Autoencoders II

AW

Lecture Overview

1. **Recap**

2. **Autoencoder Architecture**

3. **SVD: Detailed Example**

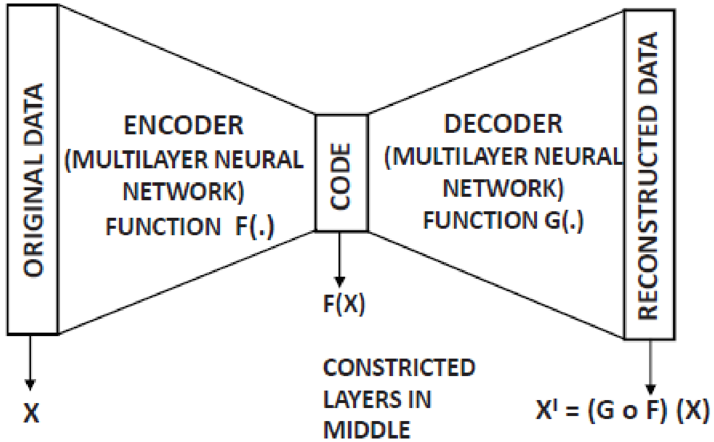
4. **Nonlinear and Deep Encoders**

Compression

- Unsupervised models are closely related to compression because compression captures a model of regularities in the data.
- Learning short feature representation implies compression
  - Generative models represent the data in terms of a compressed parameter set.
  - Clustering models represent the data in terms of cluster statistics.
  - Matrix factorization represents data in terms of low-rank approximations (compressed matrices).
- An autoencoder provides a compressed representation of the data.

Encoder and Decoder

- Reconstructing the data might seem like a trivial matter by simply copying the data forward from one layer to another.
- Not possible when the number of units in the middle are *constricted*.
  - Autoencoder is divided into *encoder* and *decoder*
  - Encoder provides compressed representation of data – code that is output $\vec{h}$ encoder hidden layer

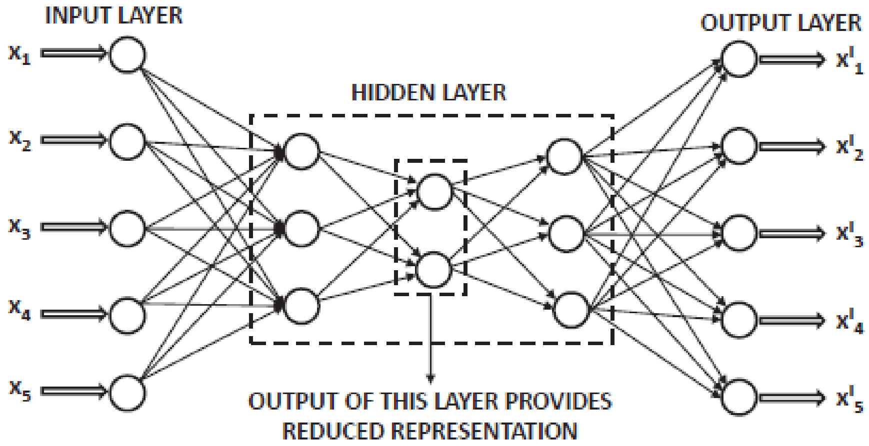


Lecture Overview

1. Recap
2. **Autoencoder Architecture**
3. SVD: Detailed Example
4. Nonlinear and Deep Encoders

Input and Output in Autoencoder

- All neural networks work with input-output pairs.
  - In a supervised problem, the output is the label.
- In the autoencoder, the type of output values are the same as inputs: *replicator neural network*.
  - The loss function penalizes a training instance depending on how far it is from the input (e.g., squared loss).



Basic Structure of Autoencoder

- It is common (but not necessary) for an M-layer autoencoder to have a symmetric architecture between the input and output.
  - The number of units in the k^{th} layer is the same as that in the $(M - k + 1)^{th}$ layer.
- The value of M is often odd, as a result of which the $\left(\frac{M+1}{2}\right)^{th}$ layer is often the most constricted layer.
  - We are counting the (non-computational) input layer as the first layer.
  - The minimum number of layers in an autoencoder would be three, corresponding to the input layer, constricted layer, and the output layer.

Autoencoders and Dimensionality Reduction

- The autoencoders that reduce dimensionality are called undercomplete
- The number of units in each middle layer is fewer than that in the input (or output).
  - These units hold a reduced representation of the data, and the final layer can no longer reconstruct the data exactly.
  - The loss function then is the reconstruction deficiency
- This type of reconstruction is inherently *lossy*.
- The activations of hidden layers may either provide an alternative or an alternative implementation to linear and nonlinear dimensionality reduction techniques.

Autoencoders and Representation Learning

- *Overcomplete* autoencoders have the number of units in hidden layer equal to or larger than input/output layers
- There are infinitely many hidden representations with zero error
- The middle layers often do not learn the identity function, especially if the loss function is based not only on replication but also on additional constraints
- Specific properties on the redundant representations can be enforced by adding constraints/regularization to hidden layer(s)
  - Training with stochastic gradient descent is itself a form of regularization.
  - One can learn sparse features by adding sparsity constraints to hidden layer.

Lecture Overview

1. Recap
2. Autoencoder Architecture
3. SVD: Mock Example
4. Nonlinear and Deep Encoders

Matrices Multiplied by their Transpose

- For any matrix D of dimension $n \times d$ consider square symmetric matrices $D^T D$ and DD^T. If $d \neq n$ they have different dimensions. Let $n < d$. Then every eigenvalue λ of DD^T is also eigenvalue of $D^T D$ and vice versa because
  - If $\lambda\vec{x} = DD^T\vec{x}$ then $\lambda(D^T\vec{x}) = D^T(\lambda\vec{x}) = D^T(DD^T\vec{x}) = D^T D(D^T\vec{x})$
  - If $\lambda\vec{x} = D^T D\vec{x}$ then $\lambda(D\vec{x}) = D(\lambda\vec{x}) = D(D^T D\vec{x}) = DD^T(D^T\vec{x})$
- Thus the set of eigenvalues of $D^T D$ and the set of eigenvalues of DD^T are the same, all eigenvalues are real and nonnegative – matrices are symmetric, but
  - multiplicities of eigenvalues are different because symmetric matrices are diagonalizable so multiplicities of $D^T D$ add to d while multiplicities of DD^T add to n
  - eigenvectors are clearly different – belong to different spaces.

SVD and Truncated SVD

- Let P be matrix composed of orthonormal eigenvectors of $D^T D$ and Q orthonormal matrix of DD^T
- Let $n \times d$ matrix Λ be diagonal with d diagonal values being shared eigenvalues of $D^T D$ and DD^T ordered from biggest to smallest and the rest of the values 0. Let then $\Sigma = \sqrt{\Lambda}$ square root taken elementwise.
- Assuming that P and Q are ordered with respect to eigenvalues in Σ we can write $D = Q\Sigma P^T$
- Truncated SVD: only the k column vectors of Q and k row vectors of P^T corresponding to the k largest singular values in Σ are calculated. The rest of the matrix is discarded.

Truncated SVD

- Truncated SVD: only the k column vectors of Q and k row vectors of P^T corresponding to the k largest singular values in Σ are calculated. The rest of the matrix is discarded.
- Any matrix of $n \times d$ matrix can be approximately written as $D \approx Q\Sigma P^T$ where Q, Σ, and P are $n \times k, k \times k$, and $d \times k$ matrices, respectively, such that P, Q have orthonormal columns and Σ is diagonal.
- Optimal truncation (retaining k largest singular values) minimize the Frobenius norm $||D - Q\Sigma P^T||_F$ (root-squared sum of residual entries in $D - Q\Sigma P^T$).
  - The value of *k* is typically much smaller than $\min\{n, d\}$.
  - Setting *k* to $\min\{n, d\}$ results in SVD (zero-error decomposition).

Relaxed SVD

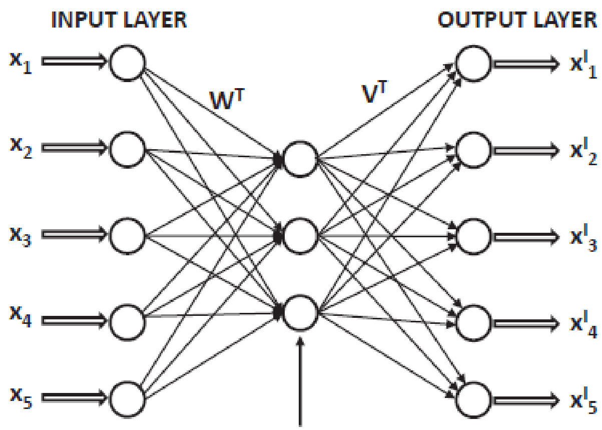
- Two-way Decomposition: Find and $n \times k$ matrix U, and $d \times k$ matrix V so that $||D - UV^T||_F^2$ is minimized.
  - Property: At least one optimal pair U and V will have mutually orthogonal columns (but non-orthogonal alternatives will exist).
  - The orthogonal solution can be converted into the 3-way factorization of SVD.
- In the event that U and V have non-orthogonal columns at optimality, these columns will span the same subspace as the orthogonal solution at optimality.

Reduced Representation= Dimension Reduction

- Any matrix factorization is a dimensionality reduction technique

$$D \approx UV^T$$

  - The n rows of D contain the n training points.
  - The n rows of U provide the reduced representations of the training points.
  - The k columns of V contain the orthogo basis vectors



In the architecture with one hidden layer:

- the rows of the matrix D are input to encoder.
- The activations of hidden layer are rows of U and the weights of the decoder contain V.
- The reconstructed data contain the rows of UV^T.

Why is this SVD?

- If we use the mean-squared error as the loss function, we are optimizing $||D - UV^T||_F^2$ over the entire training data.
  - This is the same objective function as SVD!
- It is possible for gradient-descent to arrive at an optimal solution in which the columns of each of U and V might not be mutually orthogonal.
- Nevertheless, the subspace spanned by the columns of each of U and V will always be the same as that found by the optimal solution of SVD.

Provable Facts

- The optimal encoder weight matrix W will be the pseudoinverse of the decoder weight matrix V if the training data spans the full dimensionality.

$$W = (V^T V)^{-1} V^T$$

  - If the encoder and decoder weights are tied $W = V^T$, the columns of the weight matrix V will become mutually orthogonal.
  - Easily shown by substituting $W = V^T$ above and post-multiplying with V to obtain $V^T V = I$.
  - This is exactly SVD!
- Tying encoder-decoder weights does not lead to orthogonality for other architectures, but is a common practice anyway.

Lecture Overview

1. Recap
2. Autoencoder Architecture
3. SVD: Mock Example
4. Nonlinear and Deep Encoders

Matrix Factorization (MF)

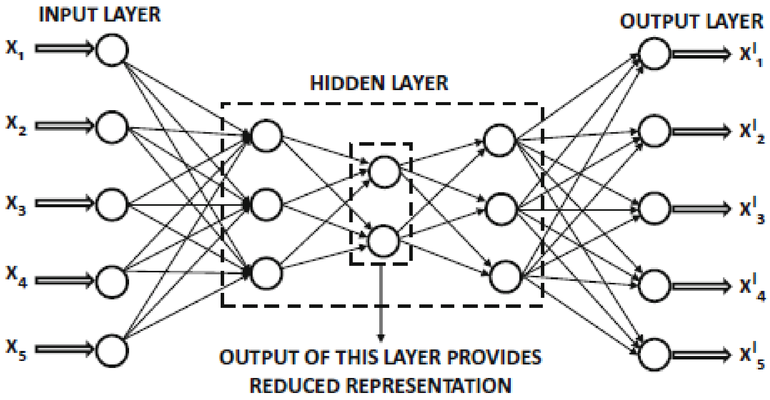
- $n \times d$ matrix data D (n inputs, d features) factorized into $D = UV^T$ where U has dimension $n \times p$ is a representation of data
- *kernel matrix factorization* nonlinear autoencoders:

Example. Shallow feature extraction NN:

- hidden layer with sigmoid and output linear.
- Input-to-hidden matrix W^T; hidden-to-output V^T
- Then output of hidden layer is $U = \text{sigmoid}(DW^T)$ notice that because sigmoid is applied elementwise U is a matrix
- Just like in linear case, we use the mean-squared error as the loss function, we are optimizing $||D - UV^T||_F^2$ over the entire training data.
- So by training we get factorization $D \approx UV^T$

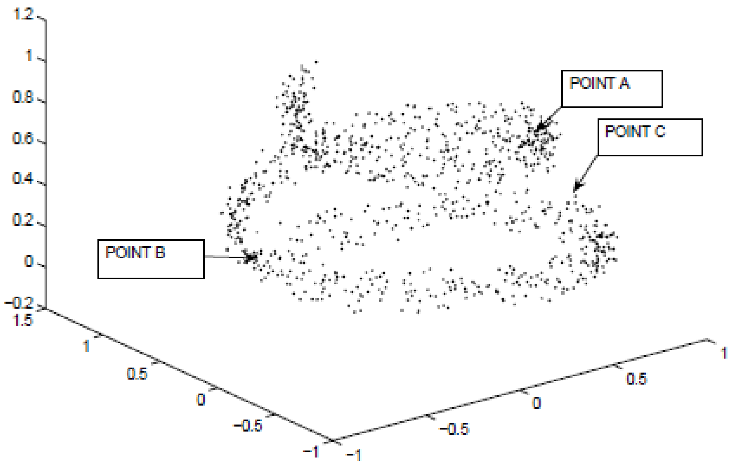
Deep encoders

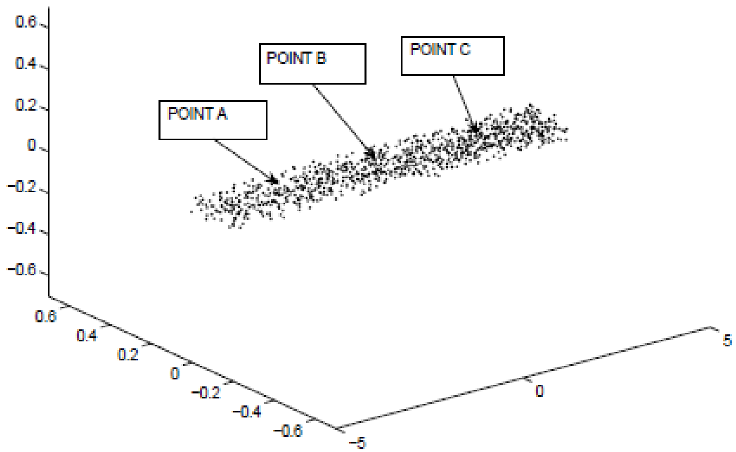
- Example is simplistic compared to what is considered typical in kernel methods, in reality multiple hidden layers are used to learn more complex forms of nonlinear dimensionality reduction



- Deep non-linear encoders can achieve reductions that are not possible by linear methods such as SVD and PCA

Non-Linear Autoencoders





- The multiple layers provide *hierarchically* reduced representations of the data.
  - For some data domains like images, hierarchically reduced representations are particularly natural.
- Just like PCA nonlinear dimensionality reduction is also a form of manifold learning but it might map a manifold of arbitrary shape into a reduced representation.
  - Extreme reductions are often achieved e.g. 784 dimensional data to 6 dimensional with images of handwriting

Reading

- Ch. 2.5.2- 2.5.3